Learning to Classify Map Data with Cascaded VLSI Neural Network Building Block Chips

T. X Brown[1], T. Duong, S. P. Eberhardt[2], M. D. Tran, T. Daud, and A. P. Thakoor
Center for Space Microelectronics Technology
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California 91109
November 18, 1992

ABSTRACT

Paper maps are an important but unwieldy data format. To increase its utility, copious amounts of map data have been scanned into a digital map knowledge base. The next task in this knowledge base is to reduce this data to its underlying feature form suitable for analysis. The size of the task requires high speed, at least at the 60,000 pixels per second rate that data can be read from an optical disk. We describe a building-block, neural network, analog VLSI chip set comprising a fully-parallel 32x32 seven bit synaptic array, and a similar array with neurons integrated along the major diagonal. This set has a 140,000 feed forward passes pcr second processing rate, sufficient for map processing. We also describe a new learning algorithm compatible with analog VLSI. The algorithm combines dynamically evolving architecture with limited gradient descent back propagation for efficient and versatile supervised learning. To enable running of the learning al gorithm in hardware, synapse circuits were paralleled for additional quantization levels. As an initial proof it was applied to the two input parity problem. The hardware-in-the-loop learning system allocated 1–3 hidden neurons for the exclusive-or problem. For the more complicated map separates problem, hardware perforrnance compares favorably with a software simulation and two statistically based classifiers.

1. Introduction:

Map data is predominantly stored in color printed sheet maps. This has a variety of problems ranging from the transport and storage of the wealth of existing map data, as well as its rapid reproduction and manipulation for end-user products. This problem spans both civilian and military domains. As a first attempt at mitigating these problems, the Army has begun to scan the maps into a 24-bit per pixel format (1 byte each; red, green, and blue) using high-resolution scanners and storing it on CD-ROM optical disks [DMA89]. This generates about 10^8 pixels per map sheet with 24 bits per pixel. The many hundreds of thousands of map sheets overwhelm even optical storage capabilities. Furthermore, the user of the maps is not interested in the exact color of each pixel—which suffers from processing variations, aging, and extraneous, aesthetic varia-

1. Present Address: Bell Communications Research, 445 South Street, Morristown, NJ 07962
2. Present Address: Department of Engineering. Swarthmore College, S warthmore, PA 19081

tions—rather, they would prefer to be able to manipulate aspects of the data such as displaying the primary road network, or ascertaining whether a particular vector crosses any river obstacles. In short, they are interested in the map features.

As a step toward reducing the storage demand while at the same time transforming the data into a format suitable for user analysis, the pixels are classified into feature classes. This reduces the storage to log(number of feature classes) bits per pixel. For a small number of classes this reduction in data is substantial. Once classified, data manipulation is greatly simplified. A neural network is chosen for the task for two reasons. First, as described in [Lee90], neural networks generate optimal decision surfaces. Second, a key requirement of the task is to avoid any processing bottlenecks beyond the limitation of the CD-ROM access rate. This implies that the classifier must process pixels as fast as they can be read from the optical storage, that is, 60,000 pixels per-second.

The inherently parallel design of a neural network as well as the biologically proven capability of scaling to tremendous levels of parallelization using low precision, non-uniform, and unreliable components suggest a strong potential for building high-speed massively-parallel neural computers even with low-precision components. Many neural applications would benefit by the tremendous speed advantage offered by fully parallel hardware implementations, wherein each connection (synapse) and processing unit (neuron) is an independent circuit. Analog Very Large Scale Integration (VLSI) appears particularly promising since it offers higher neural and synaptic densities than most digital technologies [Ebe92], at lower power consumption. However, a primary advantage of general purpose digital architectures—programmability—is lost in fully-parallel architectures. An alternative to designing an analog VLSI chip for each neural architecture is to make building blocks arrays of neurons and synapses that can be connected and cascaded for many disparate architectures [Ebe89]. In this paper, we describe a pair of such building block chips, and two applications that use cascading in three different dimensions.

One promise of neural networks lies in their ability to 'learn' to characterize data based on limited training data. However, studies have shown that some traditional learning algorithms require a high degree of weight quantization (14-16 bits) [Hoe92, Hol90] and circuit precision[Ebe88]; therefore, new algorithms tolerant of the weight quantization, noise, and other non-ideal effects of analog technology must be developed. Several such algorithms have already been reported[e.g. And90, Ebe90]. However, the neural network community has recently begun to favor learning rules that evolve the network architecture, as opposed to older algorithms that forced the user to preselect (often by trial and error) a fixed architecture. Below, we present an architecture-evolving, VLSI-compatible learning rule that was executed on the building-block chips and tested using the 2-input parity problem. To overcome the 7-bit synaptic quantization limitation of our building-block design, synapse circuits were cascaded in an interesting way (piggy-backing synapses) to obtain 11 bits of effective quantization. Additionally, the building-block chips were also cascaded laterally to increase the array size to 64x64 high-precision synapses that fully connected the 64 neurons. These were applied to the map-separates task. Comparsons were made between the results of software simulation, the hardware using the above mentioned learning, and two theoretically optimal (but computationally slow) statistical techniques.

2. Neural Hardware:

In any neural network circuit, there are two basic components: synapse and neuron. The synapse multiplies an input signal by a stored weight value while the neuron maps the sum of several such synapse outputs via a non-linear sigmoidal transfer function. By building these simple components in fully-parallel hardware architectures, the promised speed of neural networks' parallelism is achieved.

JPL has developed a variety of such structures on cascadable building block VLSI chips capable of executing virtually any neural network architecture, from feed forward to feed back. The processing is asynchronous, analog, and fully parallel, implying high speed. With a one-microsecond neuron time constant and about 1000 synapses (weights), the chips are capable of a gigs-connection per second speed. Our synapse design is based on a static random access memory (SRAM) with 7 bits (6 bits + sign bit) of resolution having two-quadrant current multipliers [Moo90]. Externally the synapses appear as a digital memory so that a write to the appropriate location changes a synapse value. The neuron design incorporates wide-range, variable-slope, sigmoid-like gain characteristics. Measurements show that the neural network chips can complete a feed-forward pass in 7 microseconds implying a processing rate exceeding 140,000 passes per-second. This is more than twice the required processing rate for the map separates problem. Two of the building-block designs are described in detail in the following subsections.

2.1. Synapse **Chip Design**

JPL has developed a 32x32 synaptic array VLSI chip with 32 input lines and 32 output lines [Duo92]. Since voltage outputs can be distributed to many high impedance inputs and a virtual-ground node can sum the currents from many sources, cascadability is aided by coding neuron inputs as currents and neuron outputs as voltages [Ebe88]. The input lines distribute a voltage to 32 synapses in the same column, and the output currents of 32 synapses in the same row are dumped onto a common output line. The relative strength of the synapses is controlled by a global input parameter.

The basic synaptic circuit, shown in Figure 1, consists of three blocks: voltage to current (V-I) convertor; six-bit Digital Analog Convertor (DAC); and current steering circuit. The V-I convertor gain is determined by transconductance between a global voltage source V_{bias} and gate voltage V_{in} of Q_1. The current of the V-I convertor is mirrored to the six-bit DAC that includes a series of six cascaded transistors Q_3-Q_8 and a series of pass transistors DO–D5. Q_3–Q_8 are scaled 1:2:4:8:16:32 so that each represents one of six bits [Moo90]. The weight value is stored in a static memory latch; each bit of which controls one of D_0–D_5. The DAC output is the total current dumped on the common line. The sign bit controls the direction of output current via transistors D_6 and $\overline{D}_6$ in the current steering circuit. When D_6 is on and $\overline{D}_6$ off, the input current is mirrored from V_{dd} to the output line. When D_6 is off and $\overline{D}_6$ on, the input current is drawn directly from the output line.

The basic chip can fully connect 32 neurons to either the same 32 neurons for a feedback circuit or 32 other neurons for a feed-forward circuit. The cascadability of these chips allows us to expand these chips in up to three different dimensions as required by the desired neural network architecture, To expand the number of inputs, synapse chips are added to increase the number of

columns in the array and corresponding outputs are connected together (Figure 2a). Additional outputs are added similarly (Figure 2b).

In addition to increasing the size of the array, the resolution of the synapses can be increased by cascading in a third dimension. The relative strength of a synapse within a chip is determined by a global reference signal. By piggy-backing (Figure 2c) an additional lower-strength synapse chip-in effect paralleling each synapse in the array with the respective synapse in the piggy-backed chip-the weight values could be resolved beyond the inherent 7 bits of accuracy to a nominal 13 bits (12 bits plus sign). While the resulting response using all 13 bits was not linear or even monotonic, using only the 11 most significant bits, the response was linear as desired. For a higher precision application, the hardware input-output response could be measured and a look-up table to order the responses created that uses all 13 bits.

2.2. Synapse-Neuron Composite Chip Design

The synapse chip provides a flexible and powerful method for connecting neurons together. For the neurons, rather than a chip with only neurons requiring the same 32 inputs and 32 outputs as the synaptic array, we replace one diagonal of the synaptic array with 32 neurons ina second composite chip as shown in Figure 3. The neuron schematic is shown in Figure 4. It is composed of three components: a comparator, an I-V convertor, and a gain controller. The input to the neuron is via a single line carrying all the currents from the connected synapses. A negative feedback circuit forces a virtual ground potential onto the summing (input) node. The comparator is high or low depending on whether the input current is positive or negative. This in turn switches on either Q_1 or Q_2 so that the input node sinks or sources all the current from the summing line completing the virtual ground circuit. One of two parallel circuits drives the output depending on whether Q_1 or Q_2 is on. We describe just the former circuit for simplicity. Current mirrors replicate the input and invert the output so that Q_6 is off and Q_8 is sourcing the input current. Transistors Q_9 and Q_{10} are always on, sourcing a current controlled by I_{ctrl}. With two transistors sourcing current into the V_{out} node and only one sinking, channel length modulation causes the output potential to raise. The resulting input output characteristic is sigmoidal. Figure 5 shows that by adjusting I_{ctrl}, the gain of the sigmoid can be controlled over a wide range.

These chips can be cascaded with the synapse chips to form large neural networks. To test all the possible cascading dimensions, we have eight VLSI chips cascaded together. A two by two checkerboard of synapse chips and synapse-neuron composite chips form a 64 neuron fully-interconnected array. Piggy backing four additional synapse chips results in 13-bit synapses.

3. Learning with Hardware

To introduce the learning with hardware problem, we first discuss usage issues, To map a given neural network architecture onto a neuron-synapse array; all the array's synapses are set to zero; for each input and neuron in the architecture, one array neuron is assigned; if weights are given, these are scaled and quantized to match the hardware synapse resolution; and then the weights are loaded into the corresponding array synapses.

This implies several technical difficulties with which the hardware must contend. To start with, weights in the network are bounded and have finite precision. In addition, because of the

processing variances across VLSI hardware and the analog signals used, the network must contend with non-uniformities in synapse responses, non-ideal neuron transfer functions, and a variety of noise sources. Inputs for the network are generated by designated input neurons. The bias weight to each of these neurons is adjusted to provide the given input level to the network. Since the inputs to neurons are controlled by weights, they too are constrained to a bounded range and finite precision. The accuracy here is further limited since only the central, 'linear', region of the sigmoidal neuron transfer function is useful. Also, the neuron outputs are not in the range [0,1], but have some voltage range determined by their operating characteristics. This is difficult to compensate under the range and precision constraints. As a result, weights learned under an ideal model cannot simply be down-loaded into the hardware and yet still find the performance at par with the simulation results. Consequently hardware must be incorporated into the learning to capture these effects using so-called hardware-in-the-loop learning.

One challenge to executing learning with analog VLSI is finding an algorithm that is tolerant of these errors inherent to the technology [Ebe92]. Among algorithms that have been successfully applied are the Madaline III rule [And90], and a related gradient-descent based algorithm [Ebe90]. Newer learning algorithms have sought to decrease learning time and obviate the user from the need to predefine the system architecture. One such algorithm, Cascade Correlation [Fah90], starts with a perception architecture and allocates hidden neurons as needed. The new Cascade Backprop algorithm proposed here combines the resource allocating structure of Cascade Correlation with the gradient descent method described by Rummelhart et al.[Rum86].

3.1. Algorithm

The algorithm is summarized here for the case of a fixed training set. Initially, a feed-forward perceptron architecture that simply connects each input to every output is mapped onto the feedback neural array. For a given set of input-output training patterns, the pseudoinverse has the property that it minimizes the mean squared error (MSE) for this architecture assuming linear output units[Dud73]. For the perception, the mapping from inputs to outputs can be turned into a linear mapping after compensating for the non-linear sigmoid function. Therefore, the pseudoinverse is used to calculate directly the initial perception weights based on input and output patterns. Back propagation learning is then used to adjust these weights. When either the system has converged or a specified number of learning trials have been completed, the network is tested on the entire training set. If the performance is satisfactory then learning stops. If not, then a hidden neuron is added and connected via synapses between each of the input units and each of the outputs.

After this initial stage, learning continues by alternately first training both the synapses connected to a new hidden unit and the output synapses via back propagation, and then adding a new hidden neuron. Note that after training, the new hidden neuron's input weights are frozen and for the purposes of this algorithm the hidden neuron output is treated as another input. This minimizes the number of synapses that must be trained through a hidden unit. Hidden units are added until the desired level of performance is obtained.

In general, backprop requires weight values, network inputs, deviation between desired and network outputs, and neuron gradients at the current input level. A key to fast learning is efficiently generating these values. The hardware weights are digital values and are stored in parallel on the host machine. The inputs could be measured at each backprop pass. Instead, they are mca-

sured once at the beginning of the learning and stored on the host machine. The output deviation is measured directly from the hardware. The neuron gradients are obtained directly from the hardware by perturbing the bias at the neuron input and observing the perturbation at the neuron output. These choices are motivated by the slow interface between the host machine and the current test set up. Ideally each weight would be perturbed individually to generate a gradient with respect to the weights that captures any non-uniformities and discrepancies between the ideal model and the hardware, but this would greatly increase the load on the test setup interface. During back propagation, the learning rate is linearly decremented over time. The learning uses all 13 bits of precision as available, and though weight updates might occasionally be in error in magnitude and even sign, the stochastic nature of analog VLSI would eventually cause bridging of the nonmonotonicities.

3.2. Results with 2-Input Parity

As an initial demonstration, the learning hardware was applied to the two-bit parity problem (exclusive-or). After each neuron addition, 5000 back propagation trials were executed. Neurons were added until the 'true' and 'false' outputs were within 25% of the output neuron's top and bottom voltage range, respectively. Theoretically only one hidden unit is required. Over 100 trials, the algorithm added 1–3 hidden units with an average of 1.2. Figure 6 shows the hardware used in these experiments: a single 32x32 synapse-neuron composite chip with a second piggy-backed synapse chip. Simulation results have shown that Cascade Backprop consistently solves parity problems with up to 7 inputs using only 4-6 hidden units, indicating larger problems are solvable. To test this, we address a map data classification problem in the next section.

4. The Map Separates Classification Problem

The data set consists of a 305X200 pixel map fragment. A gray-scale rendition appears in Figure 7a. This data is to be classified into 7 feature classes corresponding to roads, rivers, forests, contour lines, symbols/names, man-made structures, and open areas. An analyst generated a (mining set, shown in Figure 7b, by hand classifying 3800 of the 61000 pixels. A key problem was designing feature vectors for the classification problem. Using just the color information from each pixel individually was not sufficient to generate accurate feature classification. Therefore, each pixel is classified using a window of surrounding pixels so that a pixel is classified within its local context. Analysis led to choosing a 3x3 window of pixels [Bro92]. Note that each pixel generates three features for the red, green, and blue bytes so that a 3x3 window yields 27, 8-bit input features.

Five approaches were taken to solve the map separates problem. The first two were by software simulation using standard models of neurons and synapses. One network was trained according to the Cascade Correlation algorithm, the other according to Cascade-Backprop. The next method was the hardware-in-the-loop method using Cascade Backprop on the 64-neuron, 8-chip hardware. For comparison, we used the standard statistical techniques of Bayesian unimodal Gaussian and the nearest neighbors algorithm [Dud73]. The latter method is known to be asymptotically optimal (as the number of training samples grows large). The results appear in Table 1. Gray-scale renditions of the classification output of the hardware and software neural networks appear in Figures 8a and 8b respectively. We first note that the software neural network with Cascade Correlation does as well as the nearest neighbor classifier, validating the use of a neural net-

work. The Cascade Backprop did slightly worse. As expected, the hardware the loop performance is lower. The 8% discrepancy between the hardware and the software can be explained as follows.

As outlined in Section 3, the current hardware set-up suffers from several technical difficulties. Some of these we would expect to have an effect on accuracy although mitigated somewhat by the learning. For example, consider the limited input precision. Additional software simulations show that rounding the 8-bit inputs to 4-bits reduces the accuracy by only 270 although using twice as many hidden neurons. Other systematic errors could be compensated for completely by the hardware-in-the-loop learning. But, the slow communications through the current set-up interface limits the amount of learning. Both of these deficiencies are being remedied by newer hardware.

5. Summary

Classifying map data falls within the well-studied domain of pattern recognition and classification. But its demand on speed requires novel approaches. Neural networks are known to achieve as well a classification accuracy as other classification techniques while maintaining a compact parallel representation. We described a neural network approach to map feature classification and showed through simulation that it can achieve the best accuracy possible.

Reaching the necessary speed requires massively-parallel neural network hardware. Two building-block neural network chips developed at JPL were introduced that feature asynchronous parallel analog processing, and digital SRAM-like 7-bit weight storage. One features a 32x32 synapse interconnect array, and the other features 32 fully interconnected neurons. These chips provide a basis for building arbitrary network architectures. In addition, multiple synapses can be so allocated that the effective synapse resolution is increased from 7 bits to 11 bits. As a demonstration of these capabilities, a 64 neuron fully interconnected network with 11 bit synapses was made out of eight of these chips.

Biology shows sophisticated processing even under non-ideal conditions that are outside the realm of conventional learning paradigms. Typical neural hardware has precision exceeding biological systems, but still is beyond conventional learning capabilities. A new learning algorithm for hardware was developed that allocates hidden neurons to an initial perception architecture until errors are reduced to a specified level. Simulations of the parity problem (to 7 inputs) and hardware execution of the 2-input problem suggested that the learning algorithm is robust and consistent with respect to architectures produced. On the 2-input parity problem the algorithm working with the hardware allocating 1.2 neurons on average out of a theoretical minimum of one. The neural network hardware applied to the map separates problem, demonstrated the necessary speed, exceeding the CD-ROM data rate, although its accuracy was somewhat less than that projected by simulation. This discrepancy is explained by limitations of the current test setup.

Despite the inherent noise and nonlinearities of the individual analog VLSI circuits, at a system level these neural networks gave satisfactory results. That two disparate applications could be executed using the same chip set illustrates the power and versatility of the cascadable building block approach for fully parallel hardware implementations of neural networks.

6. Acknowledgments

The research described in this paper was done by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology, and was jointly sponsored by the All Source Analysis Systems Program Office, the Defense Advanced Research Projects Agency, and the National Aeronautics and Space Administration.

7. References

[And90] D. Andes, "MRIII: A robust algorithm for training analog neural networks," Proc. IEEE/INNS IJCNN, Vol. I, pp. 533-536, 1990.

[Bro92] T. X Brown, M. D. Tran, T. Duong, T. Daud, and A. P. Thakoor, Cascaded, "VLSI Neural Network Chips: Hardware Learning for Pattern Recognition and Classification," *Simulation*, Vol. 58, No. 5, pp. 340-347, 1992.

[DMA89] *Defense Mapping Agency Product Specification for Art Digitized Raster Graphics,* DMA Aerospace Center, St. Lois Missouri, Document # PS/2DJ100, April, 1989.

[Dud73] Duda R. O., and Hart P. E., *Pattern Classification and Scene Analysis,* Wiley, New York, 1973.

[Du092] Duong, T., Eberhardt, S.P., Tran, M., Daud, T., Thakoor, A. P., "Learning and optimization with cascaded VLSI neural network building-block chips," Proc. IEEE/INNS IJCNN, Vol. I, pp. 184-189, 1992.

[Ebe88] S. Eberhardt, A. Moopenn and A. Thakoor, "Considerations for hardware implementation of neural networks," Proc. 22nd. Asilomar Conf. on Signals, Systems and Computers, Vol II, pp. 649-653, 1988.

[Ebe89] S. P. Eberhardt, T. Duong, and A.P. Thakoor, "Design of parallel hardware neural network systems from custom analog VLSI "building-block" chips," Proc. IEEE/INNS IJCNN Vol. II, p 183, 1989.

[Ebe90] S. Eberhardt and A.P. Thakoor, "A supervised learning neural network from analog VLSI hardware," Conference on Neural Networks for Computing, Snowbird, UT, 1990.

[Ebe92] S.P. Eberhardt, R. Tawel, T.X Brown, T. Daud and A.P. Thakoor, "Analog VLSI neural networks: implementation issues and examples in optimization and supervised learning," *IEEE Trans. Industrial Electronics,* Vol. 39, No. 6, 1992.

[Fah90] S.E. Fahlman and C. Lebiere, "The cascade correlation learning architecture," in *Advances in Neural Information Processing Systems II,* Ed. D.S. Touretzky, Kaufmann Publ., pp. 524-532, 1990.

[Hoe92] Hoehfield, M., Fahlman, S.E., Learning with limited numerical precision using the Cascade Correlation algorithm, To be published in *IEEE Transactions on Neural Networks*

[Hol90] Hollis, P.W., Harper, J. S., Paulos, J.J., "The effects of precision constraints in a back-propagation learning network, *Neural Computation 2,"* pp. 363-373, 1990.

[Lee90] Lee, Y., Lippman R. P., "Practical characteristics of neural networks and conventional pattern classifiers on artificial and speech problems," In *Advances in Neural Information Processing* Systems *II* ed. Touretzky, D., Morgan Kaufmann, San Mateo, CA. pp. 168-177, 1990.

[Mo090] A. Moopenn, T. Duong, & A.P. Thakoor, "Digital analog hybrid synapse chips for electronic neural networks," In: *Advances in Neural Information Processing Systems 2,* Ed: D. Touretzky, Palo Alto, CA; Morgan Kaufman Publishers; pp. 769-776, 1990.

[Rum86] D.E. Rummelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation," In *Parallel Distributed Processing: Explorations in the microstructure of Cognition. Volume I: Foundations,* D.E. Rummelhart, J.L. McClelland, and the PDP Research Group, eds. The MIT Press/Bradford Books, Cambridge, MA. 1986.

Table 1: Results of different classifiers on the map classification problem

Paradigm	Technique	Classification Accuracy
Software Neural Network	Cascade Correlation	91.2%
	Cascade Backprop	90.0%
Hardware Neural Network	Cascade Backprop	**81.9%**
Statistical Classifier	Nearest Neighbors	91.9%
	Bayesian-Unimodal Gaussian	89.8%

Figure 1: Circuit diagram of the 7-bit synapse showing the three functional blocks.

Figure 2: The three dimensions of cascading synapses: more inputs (a); more outputs (b); higher precision (c).

Figure 3: A photograph of the synapse-neuron chip showing the diagonal array of 32 neurons.

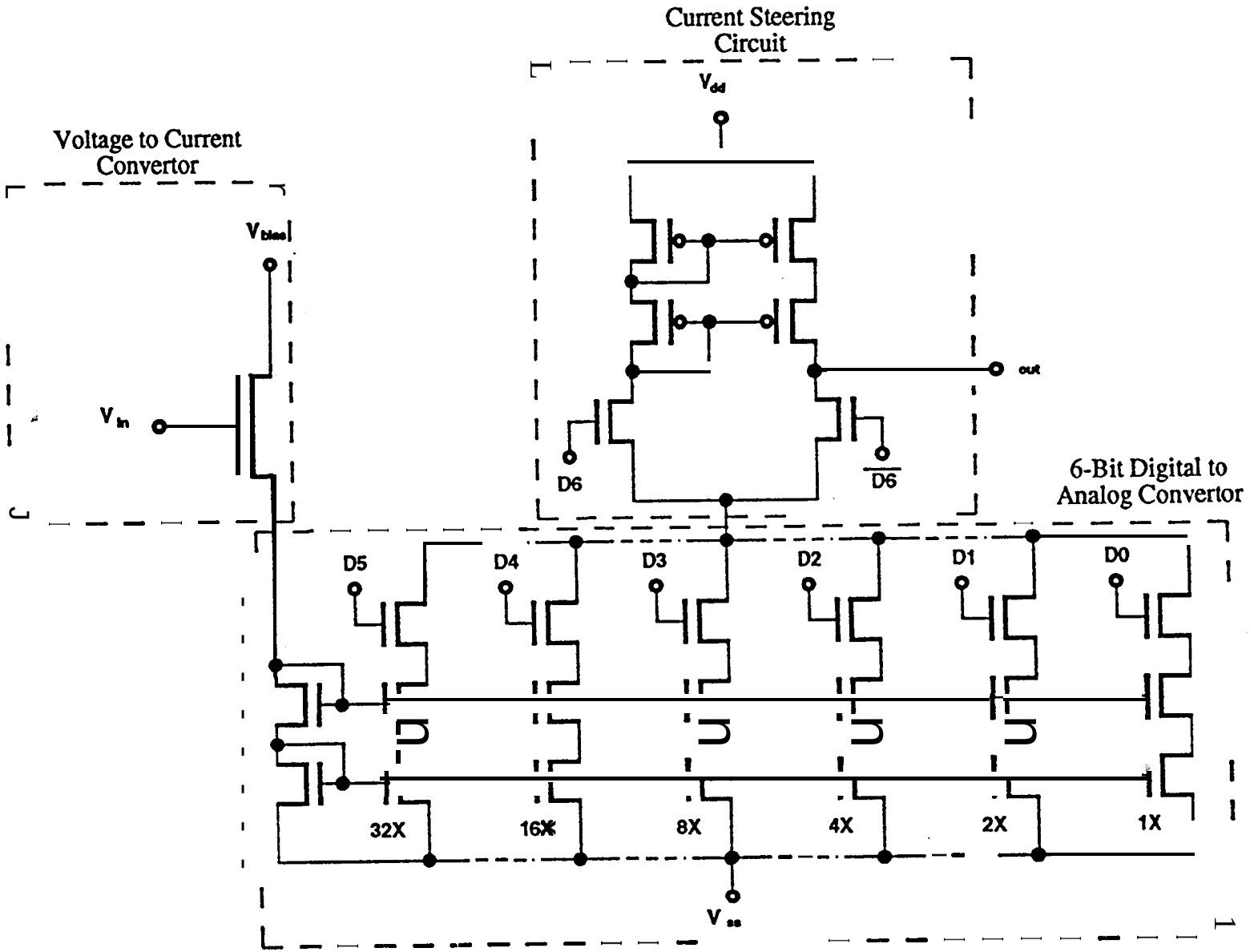
Figure 4: Circuit diagram of the variable-gain neuron.

Figure 5: Measured characteristics of a variable-gain neuron.

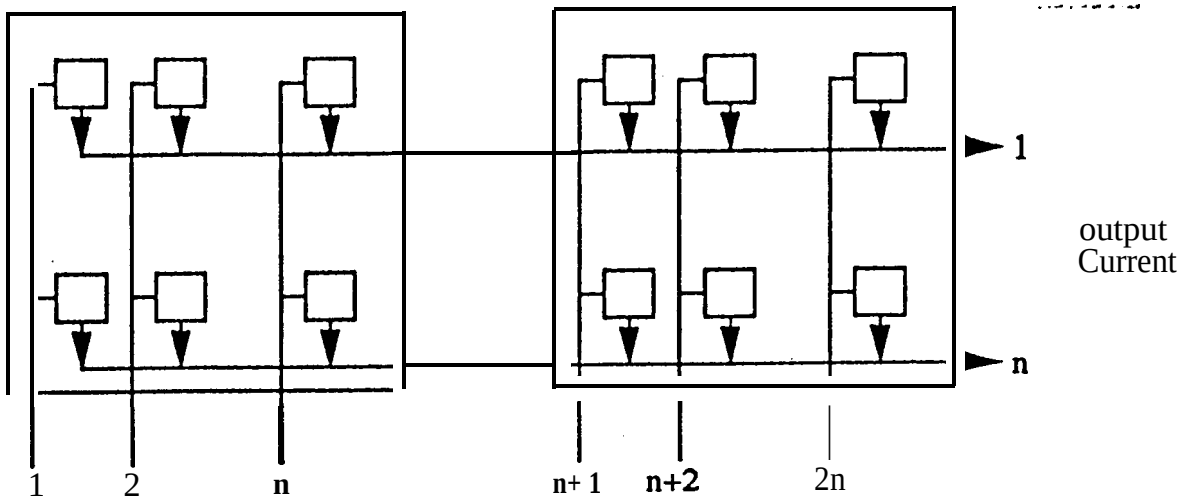
Figure 6: Photograph showing the hardware interfaced with a computer. The CRT shows the neural net architecture. The left top inset shows the result of the 2-parity problem.

Figure 7: The original map image (a), and the training data (b).

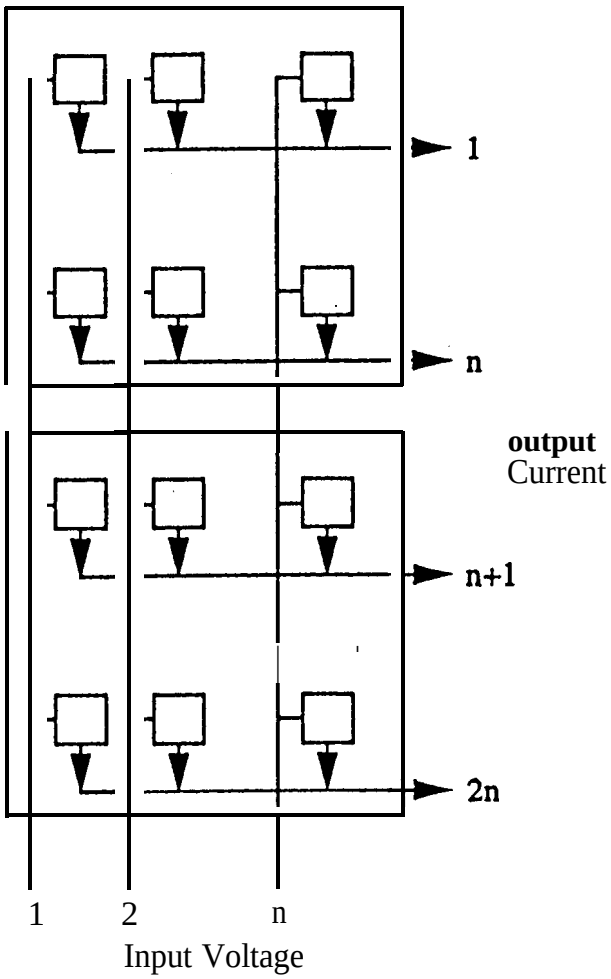
Figure 8: Classification output of the software (a) and hardware (b) neural network


Current Steering
Circuit
Voltage to Current
Convertor
V_{dd}
V_{bias}
V_{in}
out
D6
$\overline{D6}$
6-Bit Digital to
Analog Convertor
D5
D4
D3
D2
D1
D0
32X
16X
8X
4X
2X
1X
V_{ss}

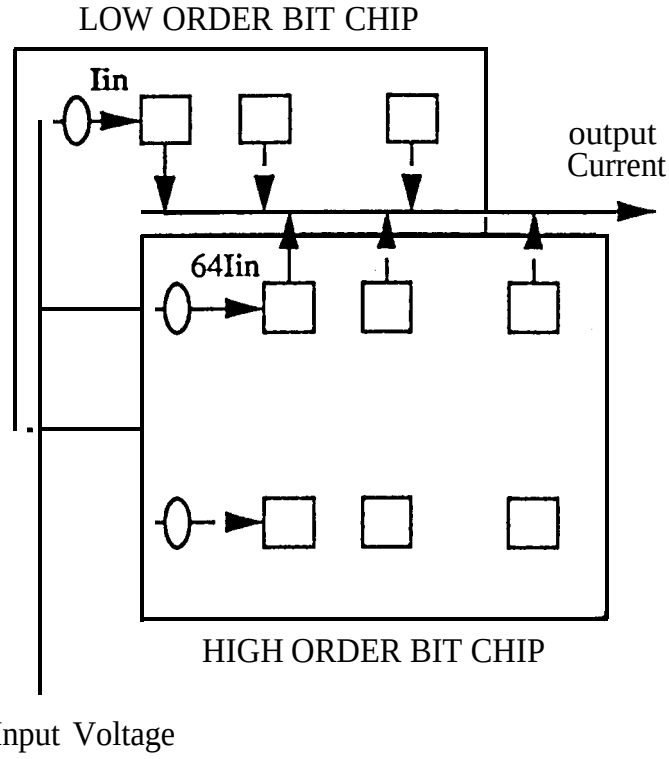
Fig


1
output
Current
n
1
2
n
n+ 1
n+2
2n
Input Voltage

(a)


1
n
output
Current
n+1
2n
1
2
n
Input Voltage

(b)


LOW ORDER BIT CHIP
Iin
output
Current
64Iin
HIGH ORDER BIT CHIP
Input Voltage

(c)

Fig 2

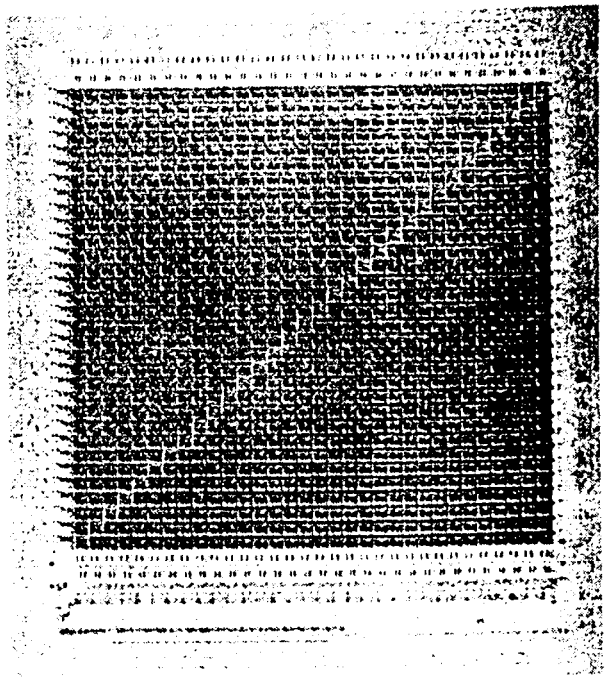


Fig 3

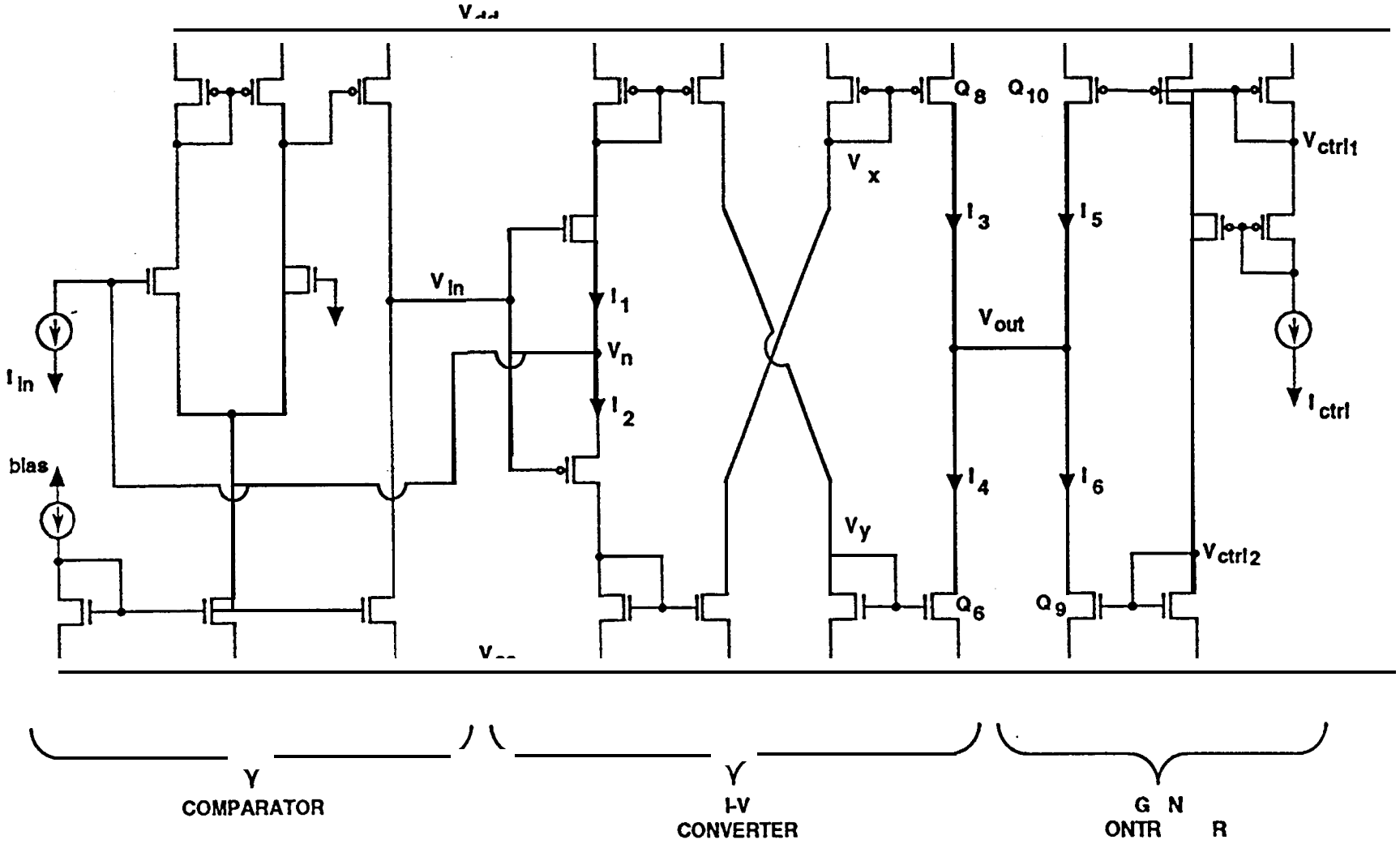

V_{dd}
Q_8
Q_{10}
V_{ctrl1}
V_x
I_3
I_5
V_{in}
I_1
V_{out}
I_{in}
V_n
I_2
I_{ctrl}
bias
I_4
I_6
V_y
V_{ctrl2}
Q_6
Q_9
V_{ss}
COMPARATOR
I-V CONVERTER
G N ONTR R

Fig. 4

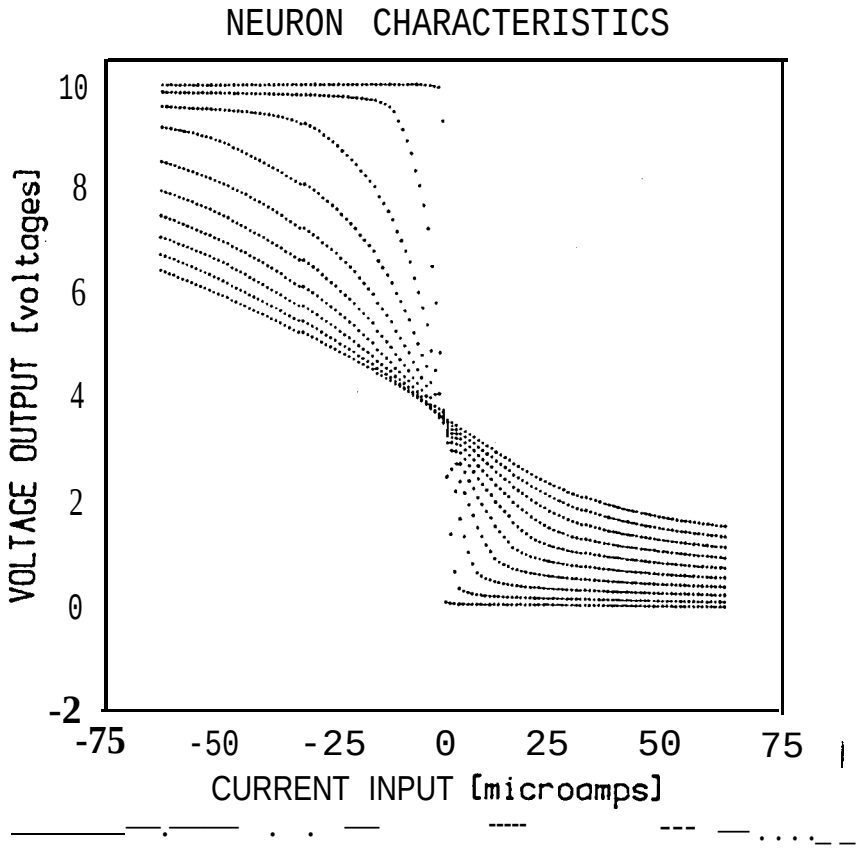

NEURON CHARACTERISTICS
VOLTAGE OUTPUT [voltages]
10
8
6
4
2
0
-2
-75
-50
-25
0
25
50
75
CURRENT INPUT [microamps]

Fig 5

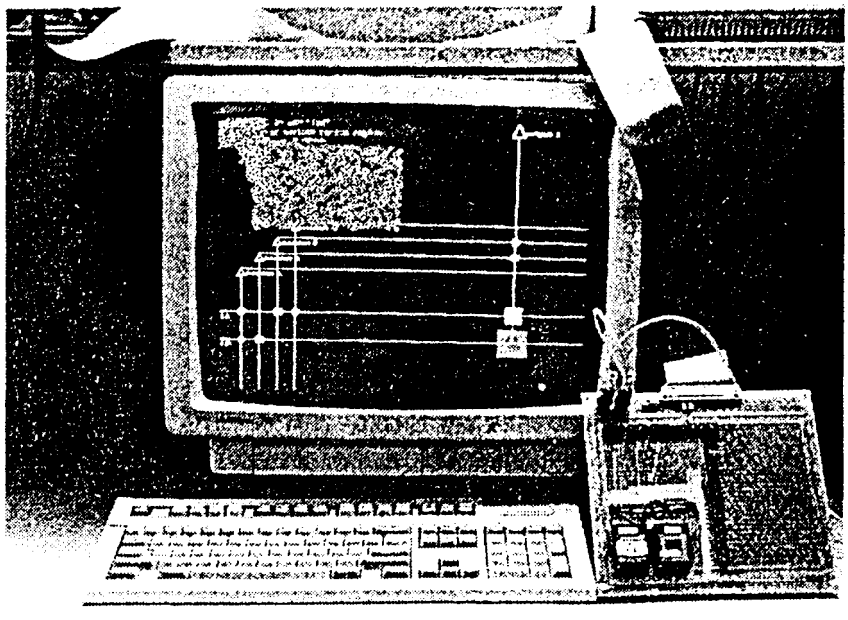


Fig 6

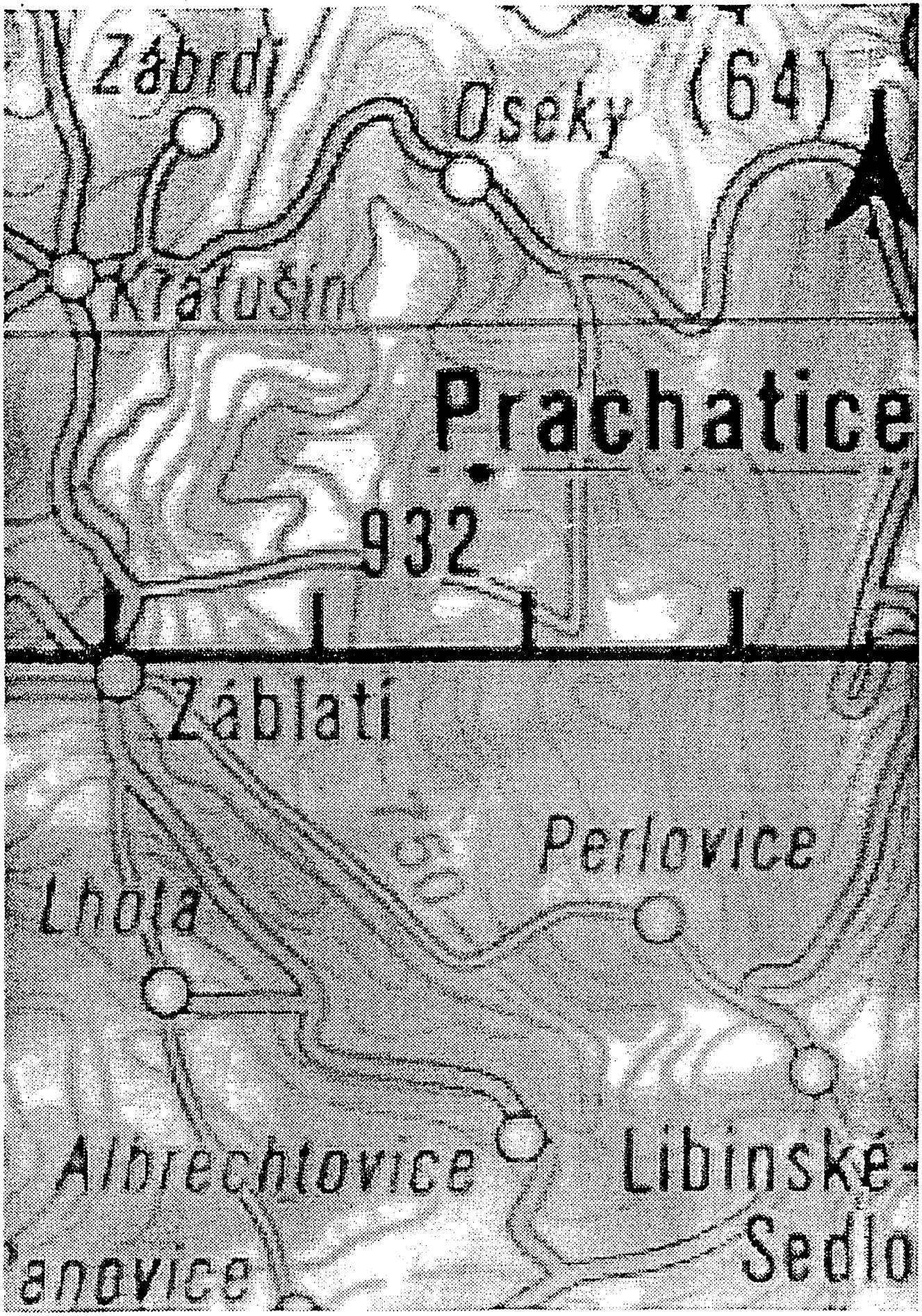

Zábrdí
Oseky
(64)
Kratušín
Prachatice
932
Záblatí
Perlovice
Lhota
Albrechtovice
Libinské-
Sedlo
anovice

Figure 3a

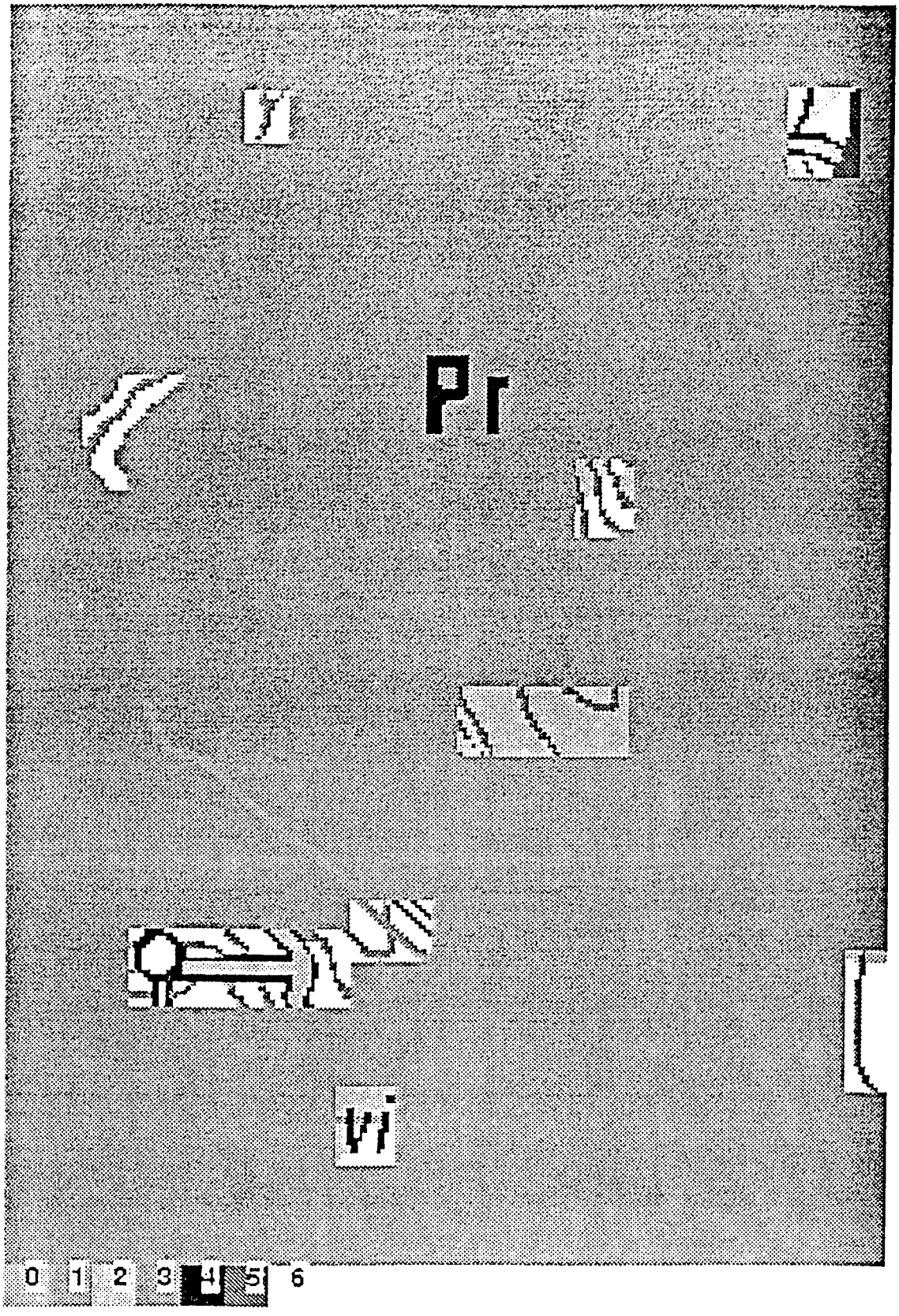


Figure 7b

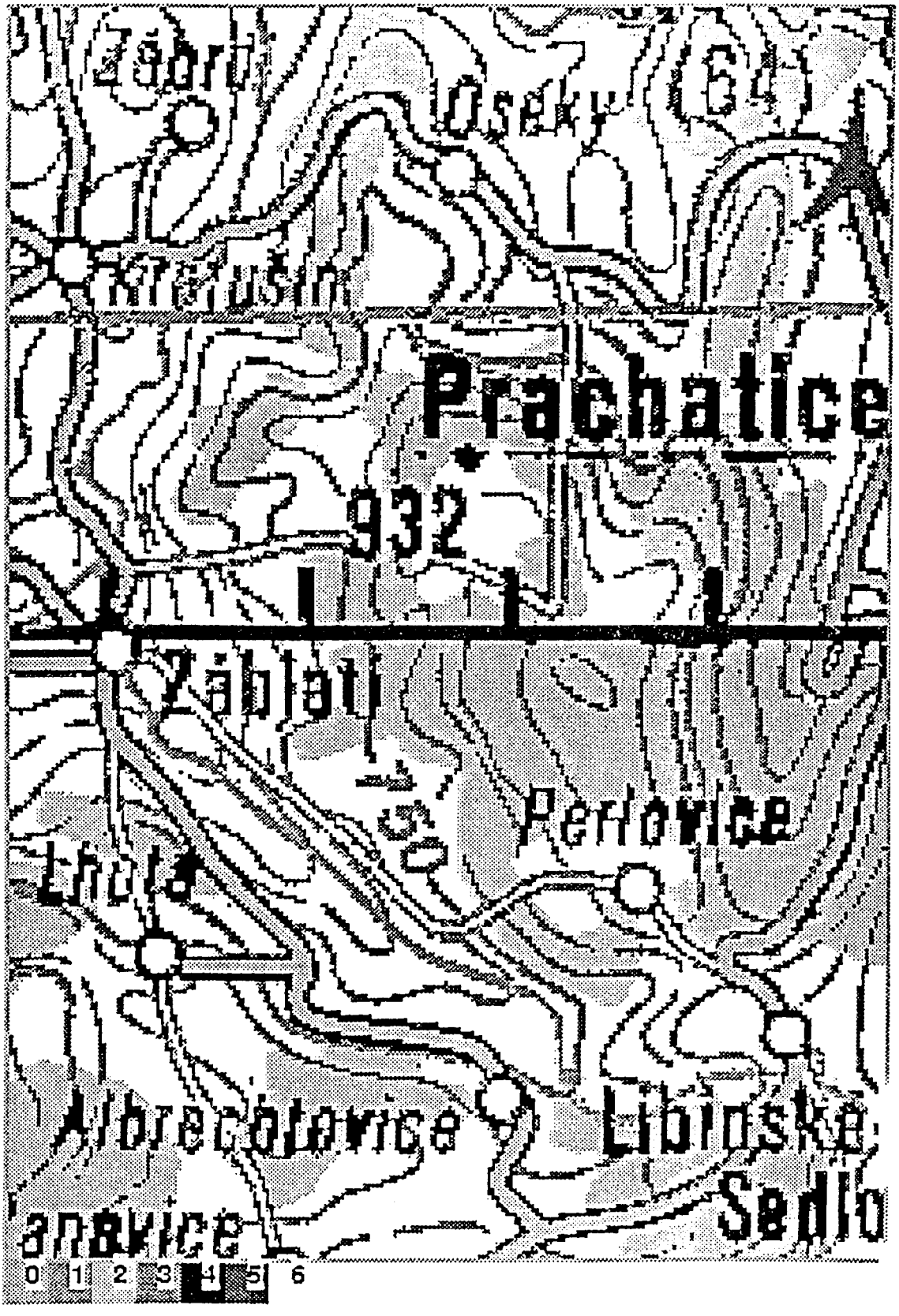


Figure 8a

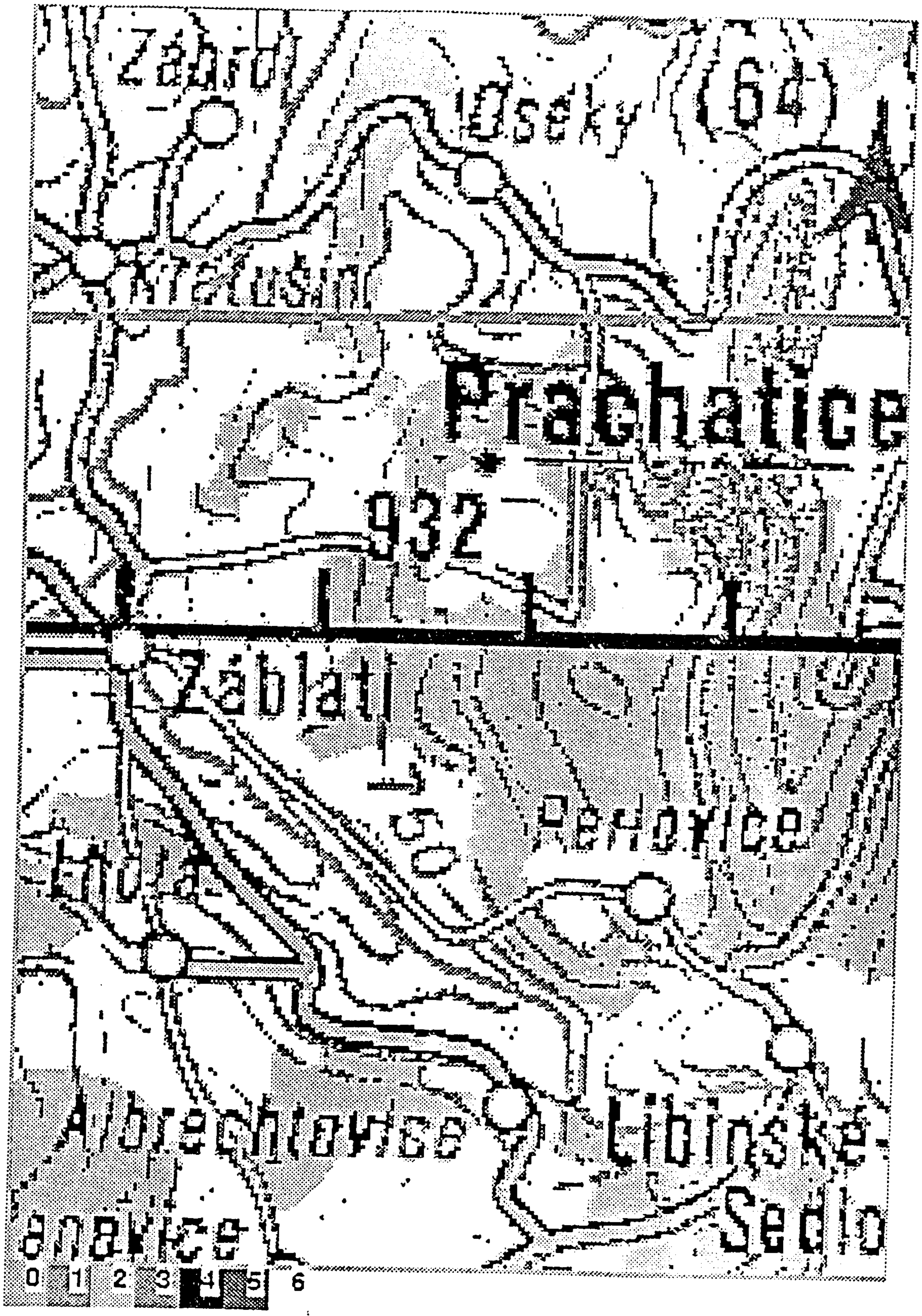

Prachatice
932
Záblatí
Albrechtovice
Libinské
Sedlo
0 1 2 3 4 5 6

Figure #8b